In [1]:
import numpy as np
import scipy.stats
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn import neighbors, datasets
from sklearn.metrics import pairwise_distances
from sklearn.utils import shuffle

# import some data to play with
iris = datasets.load_iris()
X = iris.data[:, :2]  # we only take the first two features
X = X + 0.05 * np.random.RandomState(0).randn(*X.shape)  # Add random noise since iris has exact values
y = iris.target
X, y = shuffle(X, y, random_state=0)
plt.scatter(X[:, 0], X[:, 1], c=y, cmap='rainbow')
print(X.shape)

(150, 2)
This seems odd, do we really think our method has PERFECT accuracy?

Suppose we only had 100 points for training and then received 50 new flower measurements
The above generalization accuracy estimation algorithm is known as using a train/test split

Cross validation is a better estimate of generalization accuracy
In [4]:

```python
def cv_estimate(X, k, n_splits=3):
    # Setup split indices
    split_ind = np.floor(np.linspace(0, X.shape[0], num=n_splits+1))
    # Loop over splits
    accuracy_list = []
    for split_start, split_end in zip(split_ind[:-1], split_ind[1:]):
        # Setup boolean array to select test set
        test = np.zeros(X.shape[0], dtype=bool)  # Initialize false boolean array
        test[int(split_start):int(split_end)] = True  # Set test elements to true
        # Create train and test sets
        X_train = X[~test, :].astype(int)  # ~ is used to denote "not" for all boolean values
        y_train = y[~test]
        X_test = X[test, :]
        y_test = y[test]
        # Train model for this split using X_train and y_train
        knn = SimpleKNNClassifier(X_train, y_train, k=k)
        # Compute accuracy on test split
        y_pred = knn.predict(X_test)
        accuracy = np.mean(y_test == y_pred)
        accuracy_list.append(accuracy)
        # Take mean of accuracy
    return np.mean(accuracy_list)

for k in [1, 3, 5, 7, 9, 12]:
    # n_splits is often 3, 5, 10; X.shape[0] (i.e., one split per sample) is known as Leave One Out (LOO)
    cv_acc = cv_estimate(X, k, n_splits=3)
    print(f'CV accuracy estimate for k={k} is {100*cv_acc:.1f}%')
```

CV accuracy estimate for k=1 is 74.7%
CV accuracy estimate for k=3 is 76.0%
CV accuracy estimate for k=5 is 76.7%
CV accuracy estimate for k=7 is 78.0%
CV accuracy estimate for k=9 is 77.3%
CV accuracy estimate for k=12 is 76.0%